Lösungshinweise zur Teilprüfung Software- und Internettechnologie Programmierkurs 2 Wintersemester 2004/2005

Aufgabe 1: Verständnisfragen

a)
```
do{
  scanf("Zahl: %i", &zahl);
} while (zahl!=42);
```

b)
```
a = (a^b) = 2
b = (a || c) << 3 = 8
c = a & b = 0
d = b + c = 8
```

c)
```
void minmax(int *feld, int length, int *min, int *max)
```

Beispielhafter Aufruf: `minmax(feld,5,&min,&max);`

d)
- Abfangen von Programmabstürzen und Endlosschleifen
- Falls die Watchdog-Funktion nicht regelmäßig aufgerufen wird, löst der Watchdog `RESET` aus.

Aufgabe 2: Dynamische Datenstrukturen

a)
```
knoten* vater(knoten* v, knoten* sohn){
  knoten* result;
  if (v == NULL || sohn == NULL){
    return NULL;
  } /* if */
  if (v->linkerSohn == sohn || v->rechterSohn == sohn){
    return v;
  } /* if */
  else{
    result = vater(v->linkerSohn,sohn);
    if (result!=NULL){
      /* vater gefunden */
      return result;
    } /* if */
    else return vater(v->rechterSohn,sohn);
  } /* else */
} /* vater */
```

b)
```
void freeBaum(knoten* v){
  if (v==NULL) return;
  freeBaum(v->linkerSohn);
  freeBaum(v->rechterSohn);
  free(v);
} /* freeBaum */
```

c)
```
void entferneTeilbaum(knoten* v, knoten* tb){
  knoten* vaterknoten;
  vaterknoten = vater(v,tb);
  if (vaterknoten != NULL){
    /* Referenzen auf Soehne aktualisieren */
    if (vaterknoten->linkerSohn == tb){
      vaterknoten->linkerSohn = NULL;
    }
    else vaterknoten->rechterSohn = NULL;
  } /* vater */
  freeBaum(tb);
} /* entferneTeilbaum */
```

Aufgabe 3: C-Programmierung

```
void buchstabenstat(FILE* datei){
  int zaehler['z'+1];
  int anz_buchstaben = 0;
  int zeichen;
  int i;

  /* zaehler initialisieren */
  for (i = 0 ; i < 'z' + 1 ; i++){
    zaehler[i] = 0;
  } /* for */
  /* Datei einlesen */
  for(;;){
    zeichen = fgetc(datei);
    if (zeichen == EOF){
      /* Dateiende erreicht oder Fehler */
      break;
    } /* if */
    if (!isalpha(zeichen)){
      /* zeichen ist kein Buchstabe */
      continue;
    } /* if */
    zeichen = tolower(zeichen);
    anz_buchstaben++;
    zaehler[zeichen]++;
  } /* for */

  /* Haeufigkeiten ausgeben */
  for (i=0 ; i < 'z' + 1 ; i++){
    if (islower(i)){
      printf("%c %4i mal %5.2f %%\n",
             i , zaehler[i] , zaehler[i] / ((float)anz_buchstaben)*100);
    } /* if */
  } /* for */
} /* buchstabenstat */
```

Aufgabe 4: Sensorknotensteuerung

```
.lights:
        bis.b   #0x7,&0x0029
        bic.b   #0x1,&0x0029    /* Rot */
        clr     r14
        mov     #0x0001,r15
        call    #wait
        bis.b   #0x1,&0x0029
        bic.b   #0x4,&0x0029    /* Gelb */
        clr     r14
        mov     #0x0001,r15
        call    #wait
        bis.b   #0x4,&0x0029
        bic.b   #0x2,&0x0029    /* Gruen */
        clr     r14
        mov     #0x0001,r15
        call    #wait
        bis.b   #0x2,&0x0029
        bic.b   #0x4,&0x0029    /* Gelb */
        clr     r14
```

```
    mov     #0x0001,r15
    call    #wait
    jmp     .lights
    ret
```

Aufgabe 5: MSP430-Assembler

a)
```
for(i = 0; i < lengthA + lengthB; i++)
  if (aPos == lengthA)
    c[i] = b[bPos++];
    continue;
  if (bPos == lengthB)
    c[i] = a[aPos++];
    continue;
  if (a[aPos] < b[bPos])
    c[i] = a[aPos++];
  else c[i] = b[bPos++];
```

b)
```
merge:  ;; Paramter:
        ;; r8   Startadresse Feld a
        ;; r9   lengthA
        ;; r10  Startadresse Feld b
        ;; r11  lengthB
        ;; r12  Startadresse Ergebnisfeld c

        ;; lokale Variablen:
        ;; r4   i
        ;; r5   endPosA
        ;; r6   endPosB
        ;; r7   lengthA + lengthB

        push    r4              ; Registerinhalte retten
        push    r5
        push    r6
        push    r7

        mov     r8,r5           ; Endposition Feld a initialisieren
        add     r9,r5
        add     r9,r5
        mov     r10,r6          ; Endposition Feld b initialisieren
        add     r11,r6
        add     r11,r6
        mov     r9,r7           ; r7 = lengthA + lengthB
        add     r11,r7

        mov     #0,r4           ; i=0
for_i:  cmp     r7,r4           ; (lengthA + lengthB) - i berechnen
        jeq     endfor          ; if (i==lengthA + lengthB) goto loopend
if_aPos:cmp     r5,r8           ; if (a!=endposA)
        jne     if_bPos         ; goto b_pos
        mov     @r10+,0(r12)    ; *c = *(b++)
        jmp     loopend
if_bPos:cmp     r6,r10          ; if (b!=endPosB)
        jne     if_ab           ; goto ifab
        mov     @r8+,0(r12)     ; *c = *(a++)
        jmp     loopend
if_ab:  cmp     0(r10),0(r8)    ; if (*a >=*b)
        jge     else            ; goto else
        mov     @r8+,0(r12)     ; *c = *(a++)
        jmp     loopend
else:   mov     @r10+,0(r12)    ; *c = *(b++)
loopend:add     #1,r4           ; i++
        add     #2,r12          ; c++
        jmp     for_i

endfor: pop     r7              ; Registerinhalte wiederherstellen
        pop     r6
        pop     r5
        pop     r4
        ret
```